Utilizzo del controllo Microsoft Treeview in Paradox

di Liz Woodhouse http://www.thedbcommunity.com/folk/lw.htm
traduzione di Attilio Bongiorni (a.bongiorni@libero.it)

Introduzione

Il treeviews puo' essere, in molti casi, il modo migliore di visualizzare certi tipi di dato. Tuttavia Paradox non ha un proprio oggetto di tipo TreeView, cosi' per utilizzarne uno in una form occorre usare un controllo scritto da terze parti.
Questo articolo prende in esame il controllo Microsoft Treeview che e' fornito con quasi tutte le versioni di Windows.
Vedremo cosa e' possibile fare con questo tipo di controllo, quali sono i sui limiti ed i dettagli su come realizzare quanto possibile.

Premessa

Ci sono alcune cose da sapere prima di approfondire questo articolo (vedi i punti sviluppati di seguito). Se non hai dimestichezza con i controlli ActiveX/OCX, ti raccomando di leggere i tutarial di Peter Zevenaar : http://www.thedbcommunity.com/code/activex.htm

Primo:
Quasi tutti i miei test sono stati fatti con paradox 10 e MS Treeview Control (SP4) incluso nel file MSCOMCTL.OCX. Alcuni test sono stati fatti con Paradox 8 ed MS Treeview Control 5.0 (SP2) compreso in COMCTL32.OCX e non ho notato differenze tra le due versioni.

Secondo:
MS Treeview usa "index" e "Key" come proprieta' e non c'e' assolutamente modo, con Object Pal, di leggerne i valori. Esistono inoltre altre limitazioni che vedremo in seguito in questo articolo.

Terzo:
L'MS Treeview ha soltanto una colonna. Se hai l'esigenza di implementarne uno a piu' colonne, dovrai reperire un altro OCX. Ho trovato altri OCX scritti da terzi che commentero' brevemente alla fine dell'articolo.

Quarto:
Troverai documentazione a cura della Microsoft riguardante i propri Treeview all'url:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cmctl198/html/vbobjtreeview.asp

Quinto:
Non sono riuscita a settare i font (ad eccezione di grassetto/normale), quindi se decidi di usare MS Treeview sarai fermo su Arial 12 pt.

Sesto:
Il Treeview ha una comune terminologia che e' utile conoscere. Si dichiara l'oggetto Treeview stesso esattamente col nome *Treeview*. Esso contiene una raccolta di oggetti denominati *nodes* (nodi) che, a loro volta, contengono il singolo oggetto *node*. Differenti Treeviews hanno comportamenti diversi, cambiando il modo col quale gli oggetti possono essere raggiunti dal altri oggetti (alcuni richiedono di passare attraverso l'oggetto *nodes* per accedere ad un nodo singolo, altri permettono di accedere direttamente ad un nodo specifico).
Occorre studiare la documentazione dei Treeview per scoprire esattamente come funziona ogni controllo.

Inserimento di un Treeview in una form.
Registrare in Paradox MSCOMCTL.OCX oppure COMCTL32.OCX (files che dovrebbero essere nella directory Windows\System, il secondo file contiene la versione piu' recente dell'OCX.). Questa operazione va fatta anche se il Treeview e' nella lista di paradox degli OCX. In seguito riavvia Paradox.
Crea una form in bianco, aggiungi il Treeview alla toolbar e quindi inserisci il Treeview nella form.
Setta le proprieta' a seconda delle tue esigenze: i segni + e – compaiono soltanto sul primo livello dei nodi se la proprieta' Linestyle e' = **1 – tvWRootLines**, e se lo Style e' settato su una configurazione che include "Plus Minus" symbol. E' molto consigliabile prendersi un po' di tempo per fare esperimenti con le proprietà, tanto per farsi un'idea di quali sono le possibilità e cosa funziona o meno in ambiente Paradox. Non ho ancora provato adutilizzare le immagini nel tree ma potete sempre fare i vostri esperimenti in proposito.. Per default l'utilizzatore puo' modificare il testo dei nodi, disabilita la proprieta' opportuna per evitarlo: LabelEdit = **1 – tvwManual**

Codice per leggere l'elemento selezionato.
Questa operazione e' indispensabile, visto che i parametri Key e Index non possono essere letti (Object Pal non ha la possibilita' di acquisirli). Vedremo che questa operazione richiede un codice piu' semplice di quello necessario a popolare il Treeview (riportato piu' avanti) ed e' possibile fare i primi esperimenti immediatamente visto che il Treeview alla partenza ha dei nodi di default come esempio.

Index
E' un intero modificabile, rappresenta la posizione del nodo nell'ambito dell'albero. E' a base zero e siccome l'albero puo' essere oggetto di un ordinamento esso puo' cambiare.

Key
E' un valore nascosto, definito dall'utente, univoco per ogni nodo, che non cambia e che (se potessimo leggerlo) sarebbe di estrema utilità; purtroppo cio' non e' possibile. Si puo' ovviare al problema utilizzando la proprieta' *testo* del nodo o includendo la chiave univoca nel testo usando le funzioni di manipolazione delle stringhe (come breakApart e simili) per estrarlo. NOTA: Il parametro Key serve nella sintassi del metodo add() ed e' indispensabile per relazionare i nodi padre-figlio, quindi e' possibile utilizzarlo anche per altri scopi.
Per leggere il testo dell'elemento selezionato si utilizza l'evento **click()** dell'albero (non e' un evento Paradox e compare nell'Object Explorer di Paradox con un punto rosso a fianco). Sara' opportuno usare anche l'evento keyup per rilevare la selezione anche nel caso che l'operatore utilizzi la tastiera per scorrere gli elementi del Treeview. Consiglierei l'uso dell'evento UserAction per queste azioni per non dover duplicare il codice.

```
var
  oaNode OLEAuto  ;// anche l'oggetto node e' un OCX
  stText String
endVar

;// treeTest è il nome dell'oggetto OCX sulla form
oaNode = treeTest.SelectedItem
;// SelectedItem e' una proprieta' e ritorna
;// un OLEAuto che e' il nodo selezionato

stText = oaNode.Text
;// Text e' una proprieta' del node e corrisponde al
;// testo visualizzato per il quel nodo
;// ed ora possiamo utilizzare questa proprieta' a piacimento
```

Codice per popolare l'albero
E' consigliabile sia inserire il codice in un UserAction sul metodo Open ed eseguire l'acquisizione nell'action

event dopo che l'evento Open e' stato eseguito, oppure usare un pulsante o altri eventi (non tentare di eseguire il codice direttamente nell'open).
NOTA: in questi esempi, treeTest e' il nome dell'oggetto Treeview nella form. MS Treeview ha una proprieta' chiamata nodes che ritorna una variabile OLEAuto che rappresenta l'insieme dei nodi dell'albero. La sintassi del metodo add() dell'MS Treeview e' la seguente:

nodesObject.add(li/stRelativa, liRelationShip, stKey, stText)

stText (stringa) = E' il testo che viene visualizzato dal nodo
stKey (stringa) = E' la stringa univoca che distingue il nodo aggiunto da tutti gli altri. Deve essere univoca. Raccomando di usare un campo unico o un intero (il valore del record poterbbe andare bene) in modo da poter aggiungere nodi figli a quelli di livello superiore senza problemi. Tuttavia, dai test effettuati, e' risultato che il Treeview non accetta un intero convertito a stringa, quindi bisognera' integrare la stringa con almeno un carattere alfabetico se si decide di usare un intero come chiave (vedi l'esempio).
Li/stRelative (longInt, indice di Key) = Costituisce l'indice o chiave dell'elemento gia' presente nel tree al quale si riferisce questo nodo (in inserimento). Il prossimo parametro definisce il tipo di relazione intercorrente.
LiRelationship (longInt) = Costituisce la relazione tra il nodo specificato in li/stRelative e questo nodo nuovo. I valori consentiti per questo parametro sono:

0= Primo – primo nodo in Nodes
1= Ultimo – Ultimo nodo in Nodes
2= Prossimo – Prossimo nodo in Nodes
3=Precedente – Nodo precedente in Nodes
4=Figlio – Figlio del nodo riferito in *Relative*

add() ritorna una variabile OLEAuto che gestisce il nodo appena inserito (e' possibile utilizzare questa variabile).

Esempio 1:

Semplice relazione uno – a – molti (padre/figlio) dove i figli non hanno a loro volta voci di livello inferiore a loro riferite.

```
var
  tcDetails, tcMasters TCursor
  oaNodes, oaNode OLEAuto
endVar

if eventInfo.id() = UserAction+1 then
  ;// attach all'oggetto nodes
  oaNodes = treeTest.Nodes
  ;// cancella i valori gia' esistenti
  oaNodes.clear()

  if not tcMasters.open(":TREE:MASTERS.DB") then
    errorShow("Couldn't open :TREE:MASTERS.DB")
    return
  endIf

  ;// apre la tabella relazionata a Masters
  if not tcDetails.open(":TREE:DETAILS.DB", "MasterName") then
    errorShow("Couldn't open :TREE:DETAILS.DB")
    tcMasters.close()
    return
```

```
  endIf

  ;// loop sulla tabella di primo livello
  scan tcMasters :
    ;// set range sulla tabella relazionata
    ;//per filtrare le voci relazionate alla tab. di primo livello
    if not tcDetails.setRange(tcMasters."MasterName") then
      errorShow("Couldn't set range on " + tcMasters."MasterName")
      loop
    endIf
    ;// aggiungi un record di master come nodo padre
    ;// lasciando in bianco i primi 2 parametri
    ;// ("" e [nessun argomento] sono illegali)
    ;// il nuovo nodo viene aggiunto alla fine della lista
    ;// come un nodo padre (top-level)
    ;// mastername e' sia key sia testo visualizzato
    oaNode = oaNodes.add(blank(), blank(), tcMasters."MasterName",
tcMasters."MasterName")
    oaNode.Bold = True  ;// master in grassetto
    scan tcDetails :
      ;// aggiunge i record relazionati come figli
      ;// del record in mastername
      ;// appena inserito
      ;// relationship (4) significa figlio
      ;// key e text sono  DetailName value
      oaNodes.add(tcDetails."MasterName", 4, tcDetails."DetailName",
tcDetails."DetailName")
    endScan
  endScan
  tcDetails.close()
  tcMasters.close()
endIf
```

Esempio 2:

NOTE: Il seguente esempio usa una tabella singola (di articoli e risposte ad articoli) che contiene una relazione uno – a molti di un campo del database con se stesso per mezzo di altri due campi: "atriclenumber" e' una chiave univoca che identifica ciascun record; "masteran" rappresenta il valore di *articlenumber* del corrente articolo padre oppure se stesso replicato (e' in bianco se non e' il padre). Questa relazione e' profonda n volte dato un valore n non conosciuto (non ha limiti di profondita'):

```
var
  tcMaster, tcDetail TCursor
    ;// entrambi puntano alla stessa tabella
  oaNodes OLEAuto
    ;// l'insieme degli oggetti nodes
endVar

oaNodes = treeTest.Nodes
oaNodes.clear()

if not tcMaster.open(":TREE:ARTICLES.DB") then
  errorShow("Couldn't open :TREE:ARTICLES.DB")
  return
endIf
if not tcDetail.open(":TREE:ARTICLES.DB","masteran") then
  errorShow("Couldn't open :TREE:ARTICLES.DB")
  tcMaster.close()
  return
```

```
endIf

scan tcMaster :
  ;// se l'articolo non ha padre -
  ;// viene aggiunto come l'ultimo nodo al primo livello(default)
  if isBlank(tcMaster."masteran") then
    ;// key deve essere una stringa -
    ;// ma non si puo' utilizzare string(longIntVal),
    ;// quindi aggiungo caratteri alfabetici:
    ;// text = subject :: author
    oaNodes.add(blank(),
                blank(),
                string("an",tcMaster."articlenumber"),
                tcMaster."subject" + " :: " + tcMaster."author")
  else
    ;// aggiungi un figlio
    ;// ed i records appaiono nella tabella dopo il loro padre
    ;// ma possono avere a loro volta figli, quindi vengono aggiunti
  endIf
  tcDetail.setRange(tcMaster."articlenumber", tcMaster."articlenumber")
  scan tcDetail :
    ;// aggiunge i figli (attraverso il campo 'an' del padre)
    ;// parametro (4)= figlio,
    ;// con il suo articlenumber come chiave (key)
    ;// e subject :: author come text
    oaNodes.add(string("an",tcDetail."masteran"),
                4,
                string("an",tcDetail."articlenumber"),
                tcDetail."subject" + " :: " + tcDetail."author")
  endScan
endScan

tcMaster.close()
tcDetail.close()

msgInfo("done","done")
```

Conclusioni

Visto che alcune funzioni dell'MS Treeview non sono disponibili in ambiente Paradox, in alcune situazioni ove si rendesse necessario visualizzare elementi di un database in un treeview, l'MS Treeview e' semplice da usare a condizione che se ne utilizzino soltanto le funzionalita' di base.

Altri TreeViews

Esistono anche altri treeviews disponibili oltre a quello appena trattato. Molti di essi hanno la possibilita' di gestine piu' colonne, essendo in tal modo molto piu' utili di MS Treeview. Tuttavia ho notato che ognuno di questi ha sempre qualche limitazione. Di seguito ho elencato alcuni Treeviews che ho testato con Paradox 10 .

AGGIORNAMENTO: Tony McGuire ha recentemente scoperto un problema con questi Treeview (il problema si e' presentato con SftTree/OCX 4.0 e si presume sia presente anche negli altri); a quanto pare le informazioni sulla loro licenza software viene compilata in un eseguibile. Visto che Paradox non puo' compilare un eseguibile, non si trova modo di usare il trteeview in Paradox e cosi' fino a che qualcuno non trovera' un sistema per ovviare al problema oppure un'altro Treeview piu' affine a Paradox saremo obbligati a scqgliere il

Microsoft Treeview.

SftTree/OCX 4.0 by Softel vdm, Inc - E' il piu' raccomandabile essendo il solo che ha le colonne controllabili.

- Multi colonna
- Viene fornito un buon file di Help
- Alcune sintassi non funzionanti con una normale sintassi ObjectPal, possono funzionare con una chiamata
- I font non possono essere cambiati
- Non ha un parametro Key, ma ha un "Itemdata" che e' fondamentalmente la stessa cosa
- Non avendo la Key, per relazionare un record ad un altro si utilizza la proprieta' del padre FindItemData (funziona e sembra anche veloce)

FlyTreeX by Imca System

- Non sono stata in grado di accedere alle colonne in altro modo se non dalla prima - la sintassi , riportata dalla documentazione e da una e-mail dell'assistenza ritorna un errore: "Error accessing the method..."
- La documentazione e' difficile da capire (L'Inglese non e' il loro linguaggio d'origine)
- Non c'e' modo di rendere effettiva la relazione padre-figlio se gli elementi non sono inseriti in ordine quindi i due esempi riportati sopra (1 e 2) dovrebbero essere implementati in maniera totalmente diversa per costruire il Tree.
- Questo Tree ha una proprieta' “Data” che potrebbe essere usata per memorizzare dati che non vogliemo rendere visibili all'operatore, e questo non serve nei nodi relazionati, quindi occorre liberare in modo esplicito la memoria usata da questa proprieta'

TList by Bennet-Tec Information Systems, Inc.

- Con Paradox non c'e' stato modo di accedere altre colonne a parte la prima (probabilmente ha bisogno di sorgenti ADO)

ActiveTreeView and TreeViewX by Infragistics

- Ho dimenticato cosa non andava in questo Tree ma non sono riuscito ad ottenere quello che e' possibile con i SftTree OCX

AgTreeX by Brew City Software non e' multi colonne, comunque non l'ho testato.

DataTree by GreenTree Technologies Ho dimenticato cosa non funzionava in questo.

E questi sono tutti quelli che io ho provato o almeno ho preso in considerazione.

Discussioni su questo articolo qui.